به نام خدا

تمرین کامپیوتری دوم درس طراحی کامپایلر

پاییز ۱۴۰۳

فهرست مطالب

مقدمه 2

خطاهای Name Analysis 3

1- وجود دو actor با نام یکسان در برنامه 3

2- وجود دو متغیر با نام یکسان در یک Scope 4

3- تعریف دو message handler هم‌نام در یک Actor 5

4- استفاده از متغیر تعریف‌نشده 7

5- استفاده از پردازشگر پیام تعریف‌نشده 8

6- یکی بودن نام Message Handler ها با نام Actor ها 10

نکات مهم 11

مقدمه

در فاز اول پروژه، تحلیل‌گر لغوی و نحوی (lexer and parser) زبان Soact را پیاده‌سازی کردید. همانطور که در درس مطرح شد، فاز بعدی کامپایلر تحلیل‌گر معنایی (semantic analyzer) است.

همان‌طور که می‌دانیم، در اسکوپ هر message handler می‌توان از متغیرهای مربوط به actor متناظر استفاده کرد. علاوه بر این، امکان استفاده از پارامترها و تعریف متغیرهای جدید نیز وجود دارد. بنابراین، در این مرحله باید اطمینان حاصل کنیم که تمامی متغیرهای استفاده‌شده در هر اسکوپ، به‌درستی تعریف شده باشند و هیچ‌گونه تکراری در نام‌گذاری وجود نداشته باشد. در ادامه، این قواعد را با جزئیات بیشتری توضیح خواهیم داد.

در این فاز از پروژه از شما انتظار می‌رود یک name analyzer برای زبان طراحی کنید. برای این کار لازم است موارد زیر انجام شود:

- درخت نحو انتزاعی (Abstract Syntax Tree) را با قرار دادن قواعد معنایی در گرامر بسازید. توجه کنید که node-های AST به صورت کلاس‌های جاوا در پکیج main.ast.nodes در اختیار شما قرار گرفته است. برای این قسمت، توصیه می‌شود ابتدا کلاس های مربوط به ast nodes را به دقت بررسی کنید، سپس با قرار دادن قواعد معنایی مناسب، attribute های لازم برای تشکیل دادن این node ها را از گرامر حین parse استخراج کنید.
- با پیمایش AST، اطلاعات مربوط به توابع، pattern ها و متغیرها را در جدول علائم (Symbol Table) ذخیره کنید. برای این کار لازم است تا متد visit از visitor interface در ویزیتور NameAnalyzer، برای همه node ها override شود و بررسی‌های لازم برای هر node انجام شود. در صورت تشخیص خطا، از کلاس خطای مربوطه، یک instance به آرایه nameErrors اضافه کنید.

برای داشتن درک بهتر از نحوه کارکرد کد، می‌توانید در مورد visitor pattern مطالعه کنید. همچنین برای دیدن نمونه کارکرد visitor pattern، می‌توانید ویزیتور AstPrinter از پکیج main.visitor.astPrinter را بررسی کنید. این ویزیتور پیمایش preorder درخت AST را چاپ می کند و می‌توانید از آن برای debug کردن برنامه خود استفاده کنید؛ بدین صورت که مسیرهایی که در AST طی شده را مشاهده کنید.

خطاهای Name Analysis

1- وجود دو actor با نام یکسان در برنامه

در صورتی که این خطا رخ دهد، باید خطای مربوطه اعلام شده و یک نام موقت برای Actor کنونی انتخاب شود. برای نام‌گذاری مجدد Actor، از الگویی استفاده کنید که تداخلی با نام‌های از پیش تعریف‌شده در برنامه نداشته باشد.

Line:<LineNumber>-> Redefinition of actor <ActorName>

مثال:

```
1  Actor Followers {
2      %...
3  }
4
5  Actor Followers {
6      %...
7  }
```

در این نمونه کد، دو Actor با نام یکسان تعریف شده‌اند و خطای زیر ایجاد می‌شود:

```
1  Line:5→ Redefinition of actor Followers
```

2- وجود دو متغیر با نام یکسان در یک Scope

در زبان SOACT، هر متغیر باید دارای نام یکتا در scope مربوطه باشد. استفاده از دو متغیر با نام یکسان در یک scope باعث بروز خطا می‌شود. **این قانون شامل تمامی متغیرهای تعریف‌شده در یک Actor می‌شود**. همچنین باید دقت داشت که متغیرهای داخلی Actorها فقط در همان Actor دسترسی‌پذیر هستند.

Line:<LineNumber>-> Redefinition of variable <VariableName>

مثال:

```
1  msgRcv accept() {
2      int number = 3;
3      string number = "Three";
4  }
```

در این نمونه کد، در scope داخلی پردازشگر، دو متغیر با نام یکسان تعریف شده‌اند و خطای زیر ایجاد می‌شود:

```
1  Line:3→ Redefinition of variable number
```

3- تعریف دو message handler هم‌نام در یک Actor

در زبان SOACT، تعریف دو Message Handler با نام و پارامترهای یکسان در یک Actor مجاز نیست، زیرا باعث تداخل در پردازش پیام‌ها و بروز خطا در زمان کامپایل می‌شود. این زبان از overloading پشتیبانی می‌کند، بنابراین امکان تعریف message handlerهایی با نام یکسان ولی پارامترهای متفاوت وجود دارد. اما از overriding پشتیبانی نمی‌کند، بنابراین امکان تعریف message handlerهایی با نام و پارامترهای یکسان وجود ندارد.

Line:<LineNumber>-> Redefinition of Message Handler <HandlerName>

مثال:

```
1   Actor Y {
2       ...
3
4       msgRcv receiveMessage(string msgContent) {  ...  }
5
6       msgRcv receiveMessage(string msgContent) {  ...  }
7
8       msgObs FollowRequest() {  ...  }
9
10      msgObs FollowRequest(int requestNumber) {  ...  }
11  }
```

در این نمونه کد، دو service message با نام و پارامترهای یکسان (overriding) تعریف شده‌اند(receive message) و دو observe message با نام یکسان و پارامترهای متفاوت (overloading) تعریف شده‌اند (FollowRequest)، بنابراین خطای زیر ایجاد می‌شود:

```
1   Line:6→  Redefinition of Message Handler receiveMessage
```

دقت کنید که تعریف یک observe message handler با نام یکسان با service message handler، مشکلی ندارد. بنابراین کد زیر، خطایی ایجاد نخواهد کرد.

```
Actor X {
    msgRcv receiveMessage(string msgContent) { ... }

    msgObs receiveMessage(string msgContent) { ... }
}
```

4- استفاده از متغیر تعریف‌نشده

در زبان SOACT، برای استفاده از هر متغیر، لازم است ابتدا آن متغیر در محدوده (Scope) مربوطه تعریف شده باشد. اگر متغیری قبل از استفاده تعریف نشده باشد، خطای زیر تولید می‌شود:

Line:<LineNumber>-> Variable not declared

مثال:

```
1  main() {
2      int output = input1 + input2;
3  }
```

در این نمونه کد، از آنجایی متغیرهای input1 و input2، تعریف نشده‌اند، خطای زیر ایجاد می‌شود:

```
1  Line:2→ Variable not declared
2  Line:2→ Variable not declared
```

5- استفاده از پردازشگر پیام تعریف‌نشده

این خطا زمانی رخ می‌دهد که برنامه تلاش می‌کند پیامی را به یک Message Handler ارسال کند، اما آن Message Handler در کد تعریف نشده یا در دسترس نیست.

Line:<LineNumber>-> Message Handler not declared

مثال:

```
Actor Student {
    actorVars {
        String name;
        int id;
        int gpa;
    }

    Student(String name, int id) {
        this.name = name;
        this.id = id;
    }

    msgRcv recordGPA(int gpa) {
        this.gpa = gpa;
    }
}

main() {
    Student amin = Student("Amin", 11);
    amin.sendMessage("GPA: 3.8/4");
}
```

در این نمونه برنامه، اکتور Student فاقد پردازشگر پیام sendMessage می‌باشد. بنابراین خطای زیر ایجاد می‌شود:

```
1  Line:20→ Message Handler not declared
```

6- یکی بودن نام Message Handler ها با نام Actor ها

در زبان SOACT، نام Message Handler نباید با نام Actor یکسان باشد.

Line:<LineNumber>-> Message Handler name conflicts with Actor name

مثال:

```
1  Actor Teacher {
```

```
1  Line:13→ Message Handler name conflicts with Actor name
2  Line:17→ Message Handler name conflicts with Actor name
```

```
8      Teacher(String name, int id) {
9          this.name = name;
10         this.id = id;
11     }
12
13     msgRcv Teacher(int hIndex) {
14         this.hIndex = hIndex;
15     }
16
17     msgObs Teacher() {
18         print("Hey :)")
19     }
20 }
```

در این نمونه کد، دو message handler با نام actor تعریف شده‌اند. بنابراین خطاهای زیر ایجاد می‌شوند:

```
Line:13→ Message Handler name conflicts with Actor name
Line:17→ Message Handler name conflicts with Actor name
```

نکات مهم

- تمامی فایل‌ها و کدهای خود را در یک فایل فشرده به صورت studentID1_studentID2.zip آپلود نمایید.
- در رابطه با خطاهای ترکیبی، در ویدیو توضیح صورت پروژه توضیحات دقیق‌تر داده خواهد شد.
- در صورت کشف هر گونه تقلب، نمره صفر لحاظ می‌شود.
- دقت کنید که خروجی‌ها به صورت خودکار تست می‌شوند؛ پس نحوه چاپ خروجی باید عیناً مطابق موارد ذکر شده در بالا باشد. علاوه بر آن، درخت parse ساخته شده نیز مورد بررسی قرار می‌گیرد.
- بهتر است سوالات خود را در فروم درس یا در گروه اسکایپ مطرح نمایید تا دوستانتان نیز از آنها استفاده کنند؛ در غیر این صورت به مسئولان پروژه ایمیل بزنید.